Zawansowane Modelowanie i Analiza Systemów Informatycznych

(l-9) optional

Polsko-Japońska Wyższa Szkoła Technik Komputerowych
Katedra Systemów Informacyjnych
2013

Overwiev

- **Introduction to Model Driven Architecture (MDA)**
- **Concepts**
- **Overview of Current Work**
- **Promises and Challenges**
- **Conclusions**

Software engineering evolution

- **The entire history of software engineering is that of the rise in levels of abstraction,**
- **Grady Booch Vision:**

‘ Someday soon, the idea of writing an application in Java or C++ will seem as absurd as writing an application in assembler does today. And the code generated from an Executable UML model will be as uninteresting and typically unexamined as the assembler pass of a third generation language compile is today’ -

Grady Booch (born February 27, 1955) is an American software engineer, and Chief Scientist, Software Engineering in IBM Research. Booch is best known for developing the Unified Modeling Language with Ivar Jacobson and James Rumbaugh.

Go to **"The Promise, The Limits, The Beauty of Software”**
http://video.yahoo.com/watch/577305/2839970

Executable UML – xUML

(by Stephen J. Mellor, Marc J.Balcer)

- **„Executable UML is a major innovation in the field of software development. Use it to produce a comprehensive and understandable model of a solution independent of the organization of the software implementation. It is a highly abstract thinking tool that aids in the formalization of knowledge, and is also a way of describing the concepts that make up abstract solutions to software development problems.**

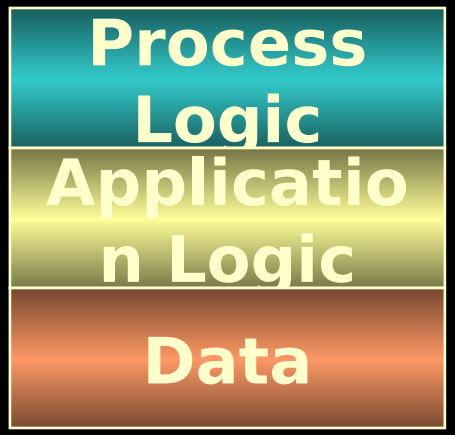
- **As a foundation for MDA, Executable UML provides the key technology for expressing application domains in a platform-independent manner.**

- **Executable UML can do more than formalize requirements and use cases into a rich set of verifiable diagrams. The models have a formal action semantics so that they are executable and testable and can be translated directly into code by executable UML model compilers".**

Software engineering evolution

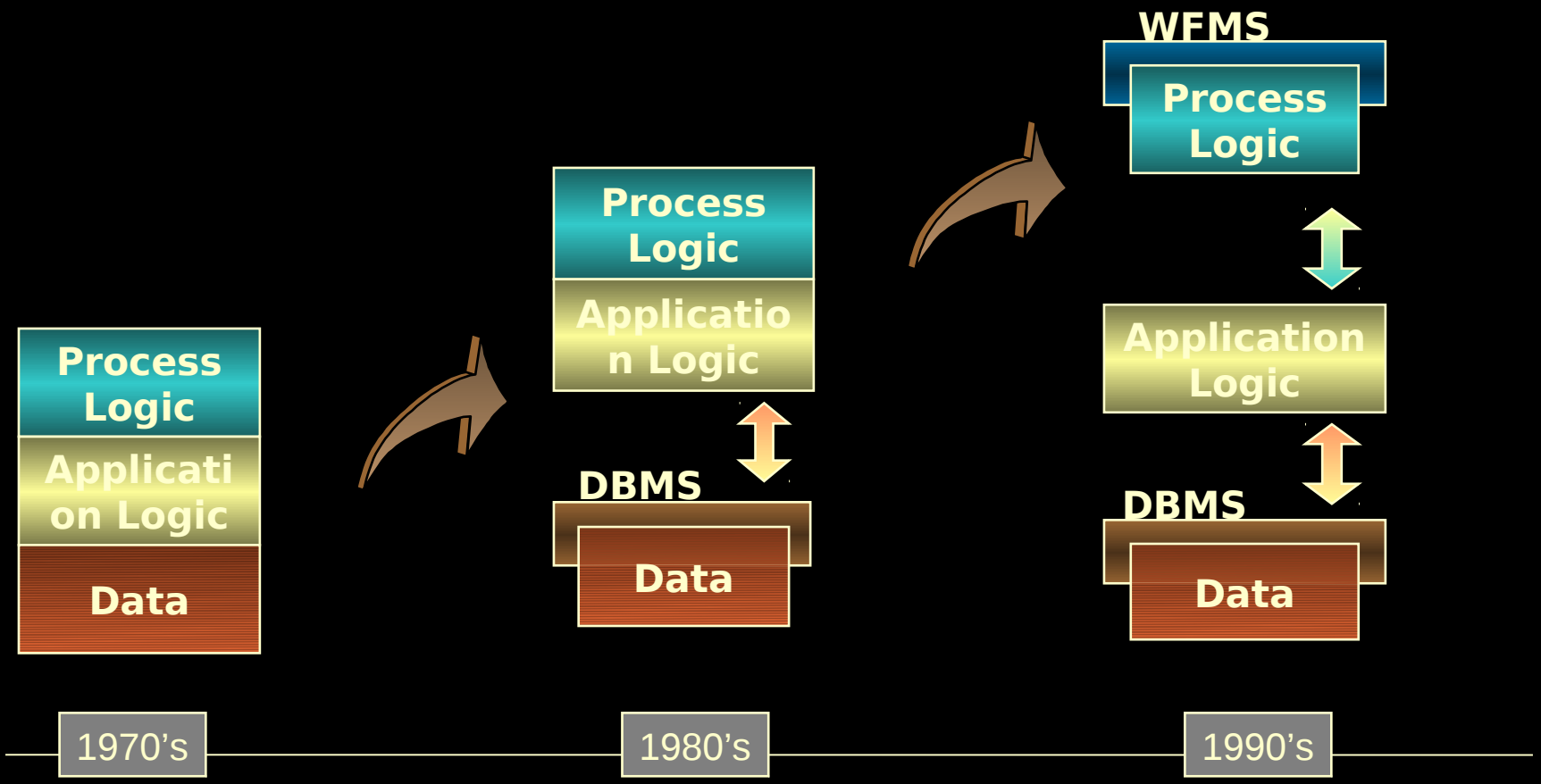
- **By the late '70s the form of Structured Programming (SP) provided a collection of good practices for writing 3GL code**
- **Followed by Structured Design (SD) and Structured Analysis (SA), both of which introduced more abstract graphical representations of programs**
- **The impact of SA/SD/SP was enormous. Defect rates dropped from 150/KLoC to 5/KLoC and great productivity improvements.**
- **There was still a problem; 60-80% of all developer effort was expended in maintaining existing software**

Software engineering evolution



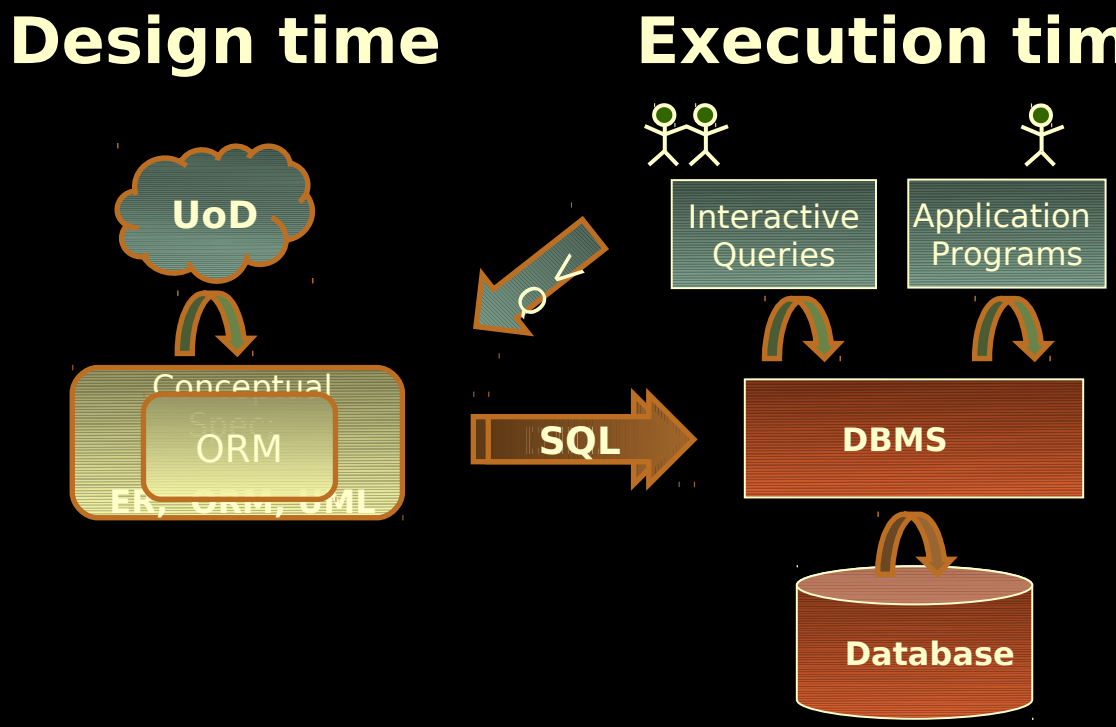
One logical block
The result was a disaster for maintainability

Software engineering evolution



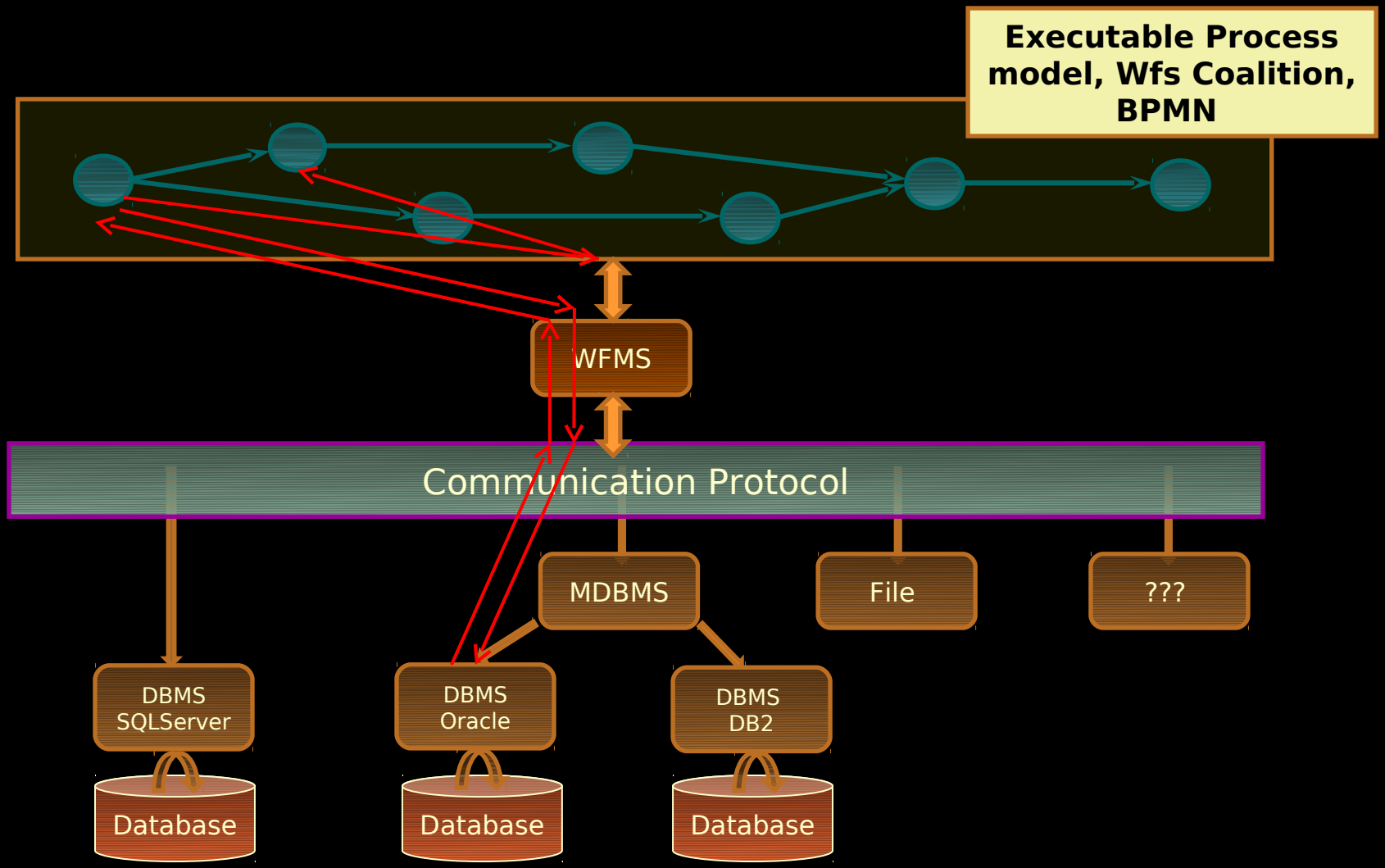
DDL and DML

executable graphical languges



Many Case Tool products use ConQuer as DDL generator

Executable Process Modeling



OMG

- **Object Management Group (OMG) is a consortium, originally aimed at setting standards for distributed object-oriented systems, and is now focused on modeling (programs, systems and business processes) and model-based standards. Founded in 1989 by eleven companies (including *Hewlett-Packard, IBM, Sun Microsystems, Apple Computer, American Airlines and Data General*), OMG tried to create a heterogeneous distributed object standard. The goal was a common portable and interoperable object model with methods and data that work using all types of development environments on all types of platforms.**
- **Today, over 800 companies from both the computer industry and software-using companies are members of OMG. Since 2000 the OMG's International Headquarters are located in Needham, Massachusetts.**

MDA as a way to go

- **In 2001 the OMG adopted the Model Driven Architecture as an approach for using models in software development - originally to help guide UML modeling**

- **MDA specifically defines three high-level model types. Each concerns a different audience, but collectively they still comprise**

one integrated model

Introduction

- ***Model Driven Architecture* (MDA) is an approach that separates a system's desired functions from its implementation on a specific technology platform, resulting in an architecture that is not tied to any one language, platform, or vendor.**
- **MDA is a platform- and vendor-neutral approach to software architecture design and development. MDA is applicable to the complete development life cycle of designing, deploying, integrating, and managing applications, using open standards such as Unified Modeling Language (UML), Extensible Markup Language (XML), XML Metadata Interchange (XMI), and Common Object Request Broker Architecture (CORBA).**

XMI

- **XMI is to enable easy interchange of metadata between UML-based modeling tools and MOF-based metadata repositories in distributed heterogeneous environments. XMI is commonly used as the medium by which models are passed from modeling tools to software generation tools as part of model-driven engineering.**
- **XMI integrates four industry standards:**

1. **XML - eXtensible Markup Language, a W3C standard.**
2. **UML an OMG modeling standard.**
3. **MOF - Meta Object Facility, an OMG language for specifying metamodels.**
4. **MOF Mapping to XMI**

CORBA

- **CORBA benefits include language- and OS-independence, freedom from technology-linked implementations, strong data-typing, high level of tunability, and freedom from the details of distributed data transfers.**
- **CORBA uses an interface definition language (IDL) to specify the interfaces that objects will present to the *outside world*. CORBA then specifies a "mapping" from IDL to a specific implementation language like C++ or Java. Standard mappings exist for Ada, C, C++, Lisp, Ruby, Smalltalk, Java, COBOL, PL/I and Python. There are also non-standard mappings for Perl, Visual Basic, Erlang, and Tcl implemented by object request brokers (ORBs) written for those languages.**

Introduction (cont)

- **MDA systems are modeled based on functions, rather than language, platform, or technology, meaning that a well-built MDA-based system can be changed or extended over time without disrupting the core infrastructure.**

- **The MDA approach potentially eases integration, shortens development time, and conserves company resources by making it possible to develop more solutions without needing more people or time.**

The basics of MDA

- **In the MDA lexicon, a *model* is a description of a system and its environment. It is often a combination of drawings and text. A *model-driven* approach is one that uses a model to direct the design, development, and maintenance of the system. It naturally follows that a *model driven architecture* is one where the architecture of the system is derived from the model of the system.**

- **MDA has three viewpoints where by a *viewpoint* we understand a technique for focusing individually on particular concerns in a system.**

The basics of MDA

- **The first MDA viewpoint is called the *Computation Independent Viewpoint* (CIV). Its role is to separate the fundamental logic of the system from the platform-specific specification. The CIV focuses on the environment and requirements of the system. In this viewpoint, the structure and implementation of the system are hidden, or possibly not yet implemented.**
- **The second viewpoint is the *Platform Independent Viewpoint* (PIV). This viewpoint focuses on the operation of the system while hiding the platform-dependent details. It may use a general-purpose, platform-independent modeling language such as UML.**

The basics of MDA

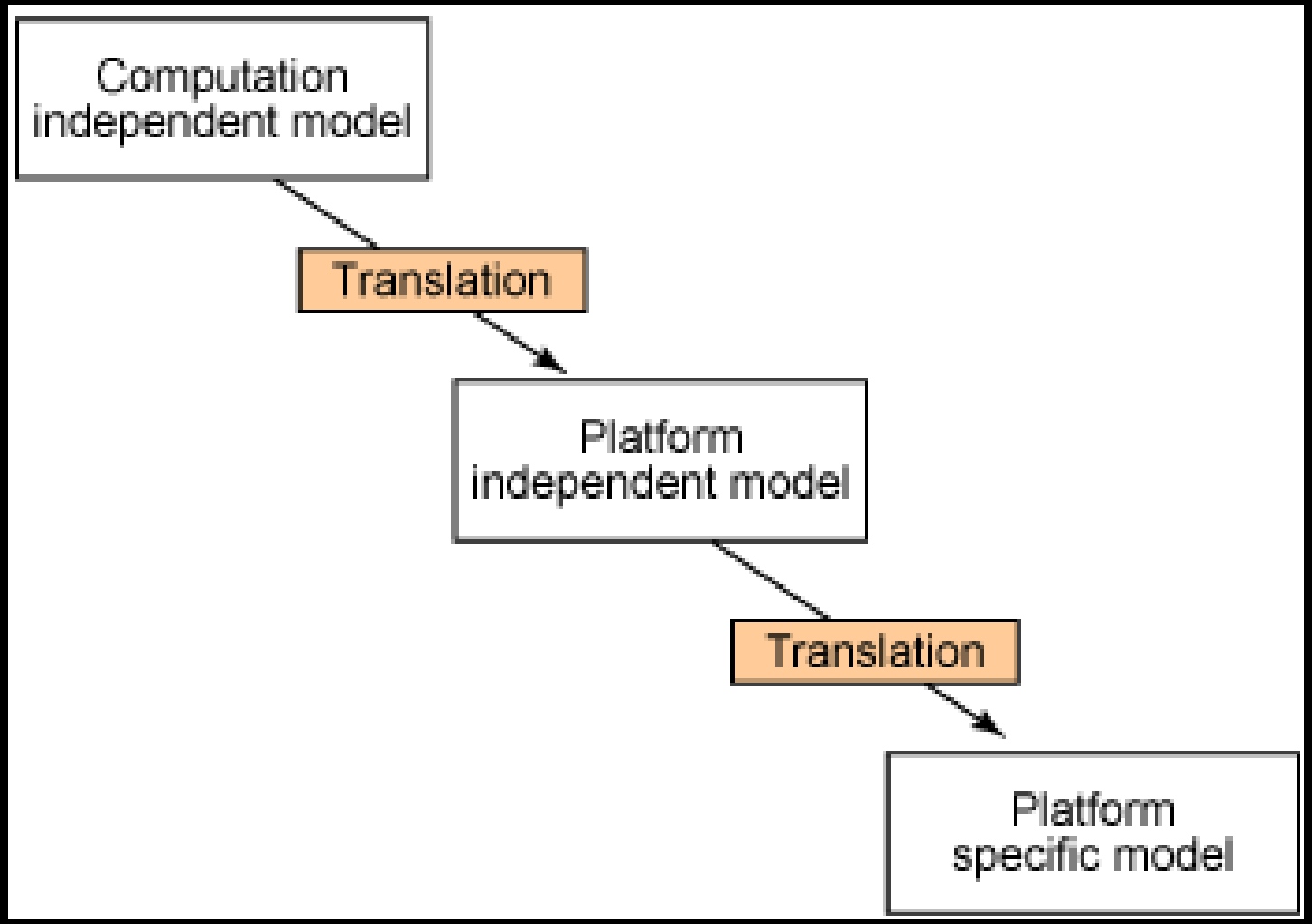
- **The third viewpoint is called the *Platform Specific Viewpoint* (PSV). This viewpoint focuses on the implementation details of a certain platform.**
- **Each of the viewpoints has its own model:**
- **The Computation Independent Model (CIM), which shows the business model of the system and is usually made by a business analyst.**
- **The Platform Independent Model (PIM), which is a model of the system functions, usually made by an architect.**
- **The Platform Specific Model (PSM), which models the implementation, on one or more platforms, of the PIM.**
- **The PIM and PSM are more important to the software .**
-

Model transformations

- **The real value of MDA lies in the fact that the CIM can be translated to a PIM by a simple mapping. Likewise, the PIM can be translated to a PSM (by a mapping), and the PSM can be translated to code.**

The key elements are the mappings and the MDA tool or tools that do the translation.

Model transformations



MDA origin

- **MDA emerged in 2002, originally to help guide UML modeling. MDA specifically defines three high-level model types: (CIM), (PIM), (PSM).**

Each concerns a different audience, but collectively they still comprise one integrated model.

An overview of a simple example

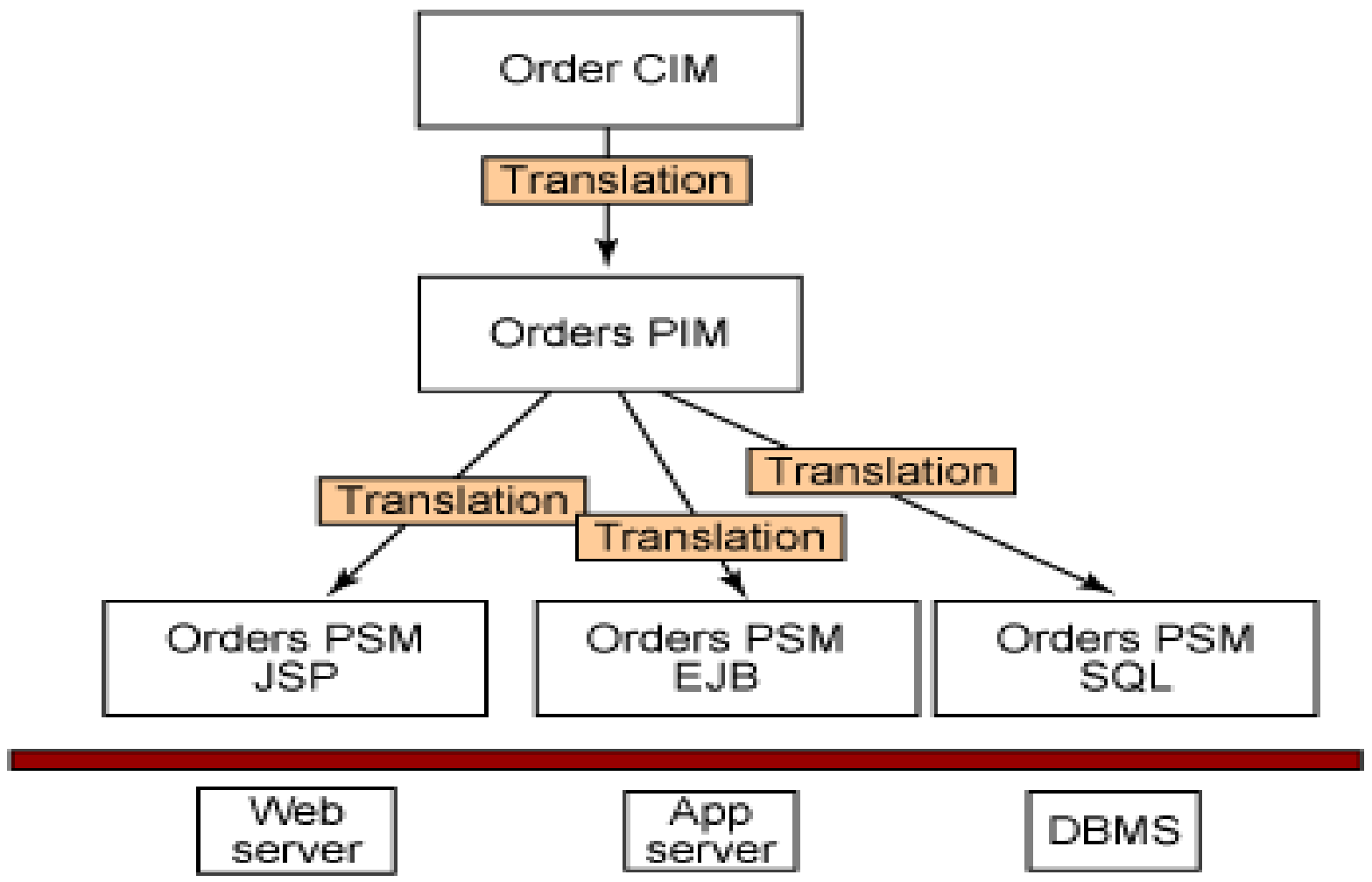
- **Think, for example, about a typical enterprise system that deals with customer orders.**

  The CIM of the Orders system shows what the system should do. The CIM would likely consist of a few high-level UML diagrams created by a business analyst. The architect would then use the translation of the CIM as a basis for creating the PIM.
- **The PIM shows the functions and structure of the system, but hides implementation-specific and platform-dependent information. The PIM consists of detailed UML diagrams that do not contain any technology-related information.**

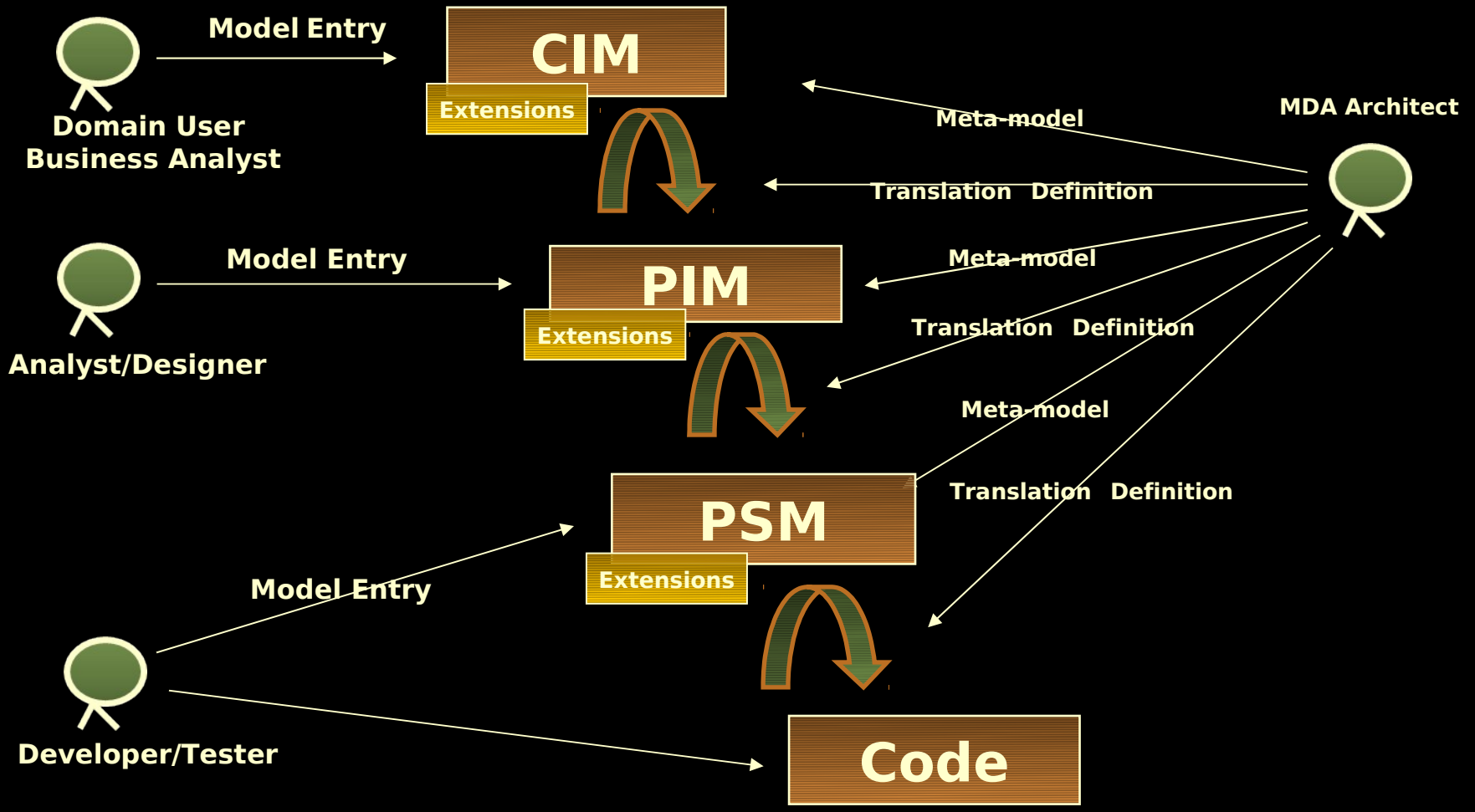
An overview of a simple example (cont)

- **The PIM is then translated, for the sake of this example, to three different PSMs: one for the database, one for the Enterprise JavaBean (EJB), and one for the JavaServer Page (JSP). The JSP will handle the user interface and, therefore, requires its own PSM (separating it from the EJB function).**
- **The next steps would be for developers to add marking to the models (prepping them for translation) and glue code to the final output.**

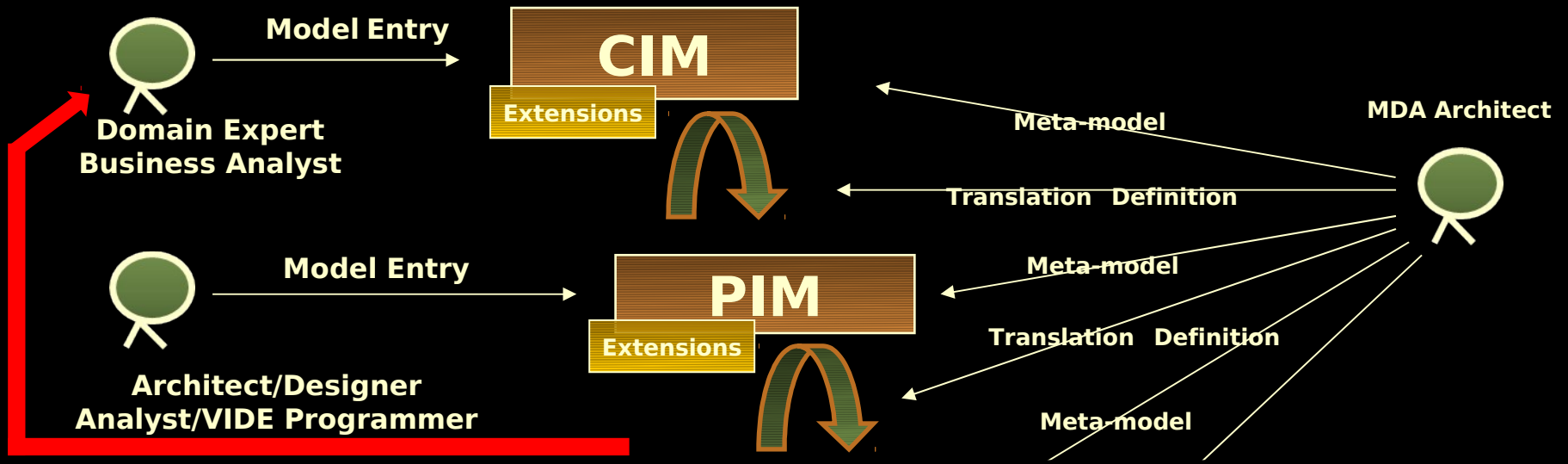
Mapping and translations for the Orders example



MDA – models stack



VIDE vision



- VIDE developes Visual and Textual languages
- VIDE provides Bevahioural modelling on PIM level
- VIDE demonstartes Automated transformations from PIM into PSM&Code

Roles

- **The CIM, aimed at business analysts, includes only the model elements necessary to describe business functionality.**

- **The PIM, for system analysts, includes additional elements describing the computational logic necessary to realize CIM functionality, but without implementation details.**

- **The PSM is aimed at system designers; it provides additional elements specific to given deployment platforms, such as Java/EJB, or C#/.NET.**

Benefits of MDA

- **MDA provides *a nice*, solid framework to let the system architecture define the models of the system before any implementation effort has begun.**

- **It also uses ready components and existing mappings to produce similar functions regardless of the platform or technologies used.**

- **Further benefits of MDA include the following:**

Benefits of MDA

- **Portability**

  After a PIM existance, it is easy to create a new PSM based on that model. Naturally, we need the mapping for the desired platform and the glue code, but the underlying model of the system will be unchanged.

- **Cross-platform interoperability**

  In addition to being able to port a system model to various implementations, one can also use a special mapping to translate a PIM into a heterogeneous PSM, in which components from multiple platforms comprise the system.

Benefits of MDA

- **Productivity**

  MDA is a highly efficient design and development approach, making it possible to get the same work done with fewer people, or to do more work with the same number of people, all without additional strain on the development team.

- **Quality**

  The incidence of human error is greatly reduced when the majority of system code is generated and derived from a single model.

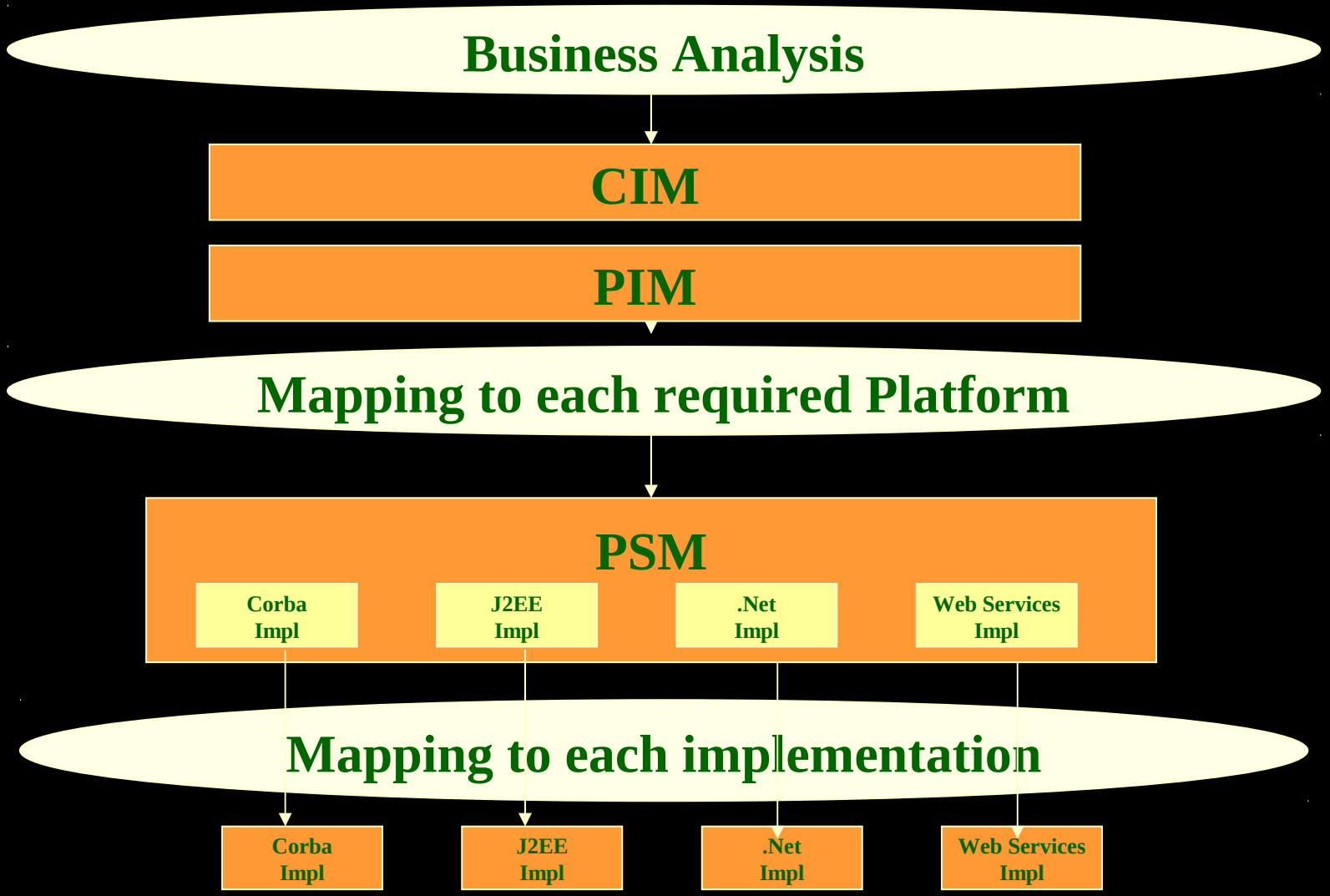
Benefits of MDA

- **Rapid inclusion of new technology**

  MDA's mapping approach makes it possible to implement a given model with new and emerging technologies, or to add newer technologies into an existing system with little strain on the development team or the core system.

- **Further benefits to this list are the benefits of platform independence, domain specificity, reduced cost, and reduced development time.**

MDA –models stack



How to construct the MDA models stack?

Customised Zachman Framework for Enterprise Architecture

http://en.wikipedia.org/wiki/Zachman_framework

← **Modelling Domains** →

Perspective	Process (function)	Application (service)	Information (data)	Organisation (resources)	Technology (infrastructure)
Contextual Vision/Goal (CIM)					
Conceptual (CIM)					
Logical (PIM)					
Physical (PSM)					

Models

- **Each entry in the Zachman EA matrix has its own;**
  - content,
  - meta-model (more specifically - the full 4 layered meta-model stack; MOF model, UML meta model, UML models, User objects),
  - reference model and
  - enterprise model.

- **The relationship between the individual models on the corresponding levels must be well designed, specified and maintained.**

- **Formally, with this only limited view on ZF, we have already 4x5x4 (perspectives, domains, 4 layered meta models) 80 models. Some are meta-models others objects models.**

Comments

- **At M2 level all 20 models may have two forms;**

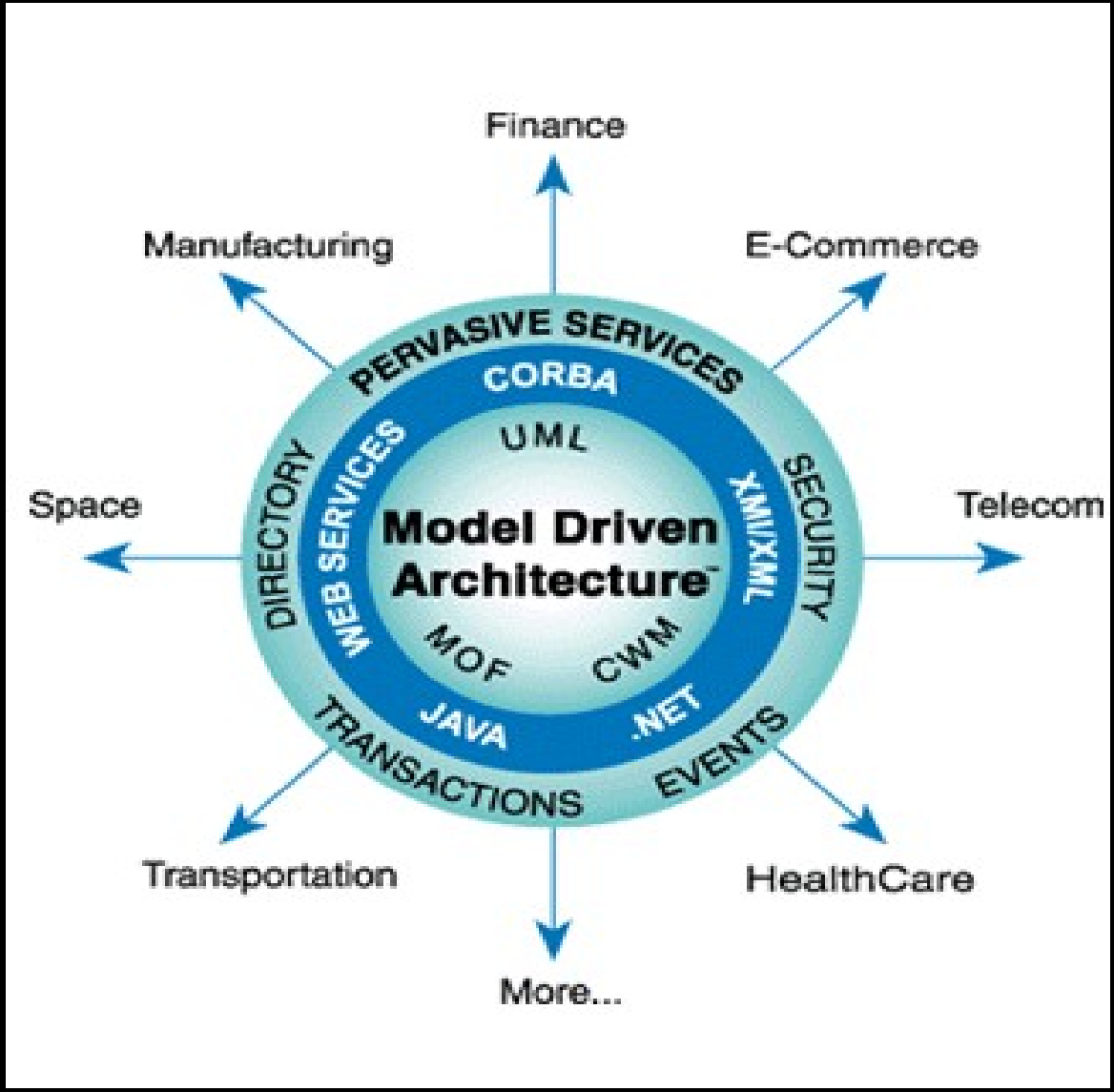
  1) abstract syntax ,

  2) specific - concrete syntax adopted by a vendor providing such modelling environment tool support.

- **The M2 models are tightly connected.**

- **Similar relationship is at other M levels.**

MDA – a perspective



MDA and UML Roots

- **Originally, UML focused on modeling the structural components of single application. But over time, OMG progressively extended UML to cover a wider range of content, including business domain models, use cases, activity flows, logical constraints, state machines, deployment packaging and service architectures.**

MDA and MOF

- **Going down the MDA's CIM-PIM-PSM route, it soon becomes clear that just extending UML is not enough.**

- **There are many other pre-existing, non-OMG notations for modeling data structures, message protocols, business processes, and so on that UML is unlikely to replace.**

- **OMG decided to integrate all these into MDA by cleverly adapting its pre-existing repository standard, the Meta Object Facility (MOF).**

MDA and SOA

- **Services Oriented Architecture (SOA) is an emerging architectural approach that seeks to develop and deploy business applications as a set of reusable, composable services.**
- **These basic concepts are not unique to SOA; some pre-existing OMG specifications address this space, including; 1) the CORBA Component Model (CCM) and**

  2) the Enterprise Distributed Object Computing (EDOC) standards. (EDOC is proposed as the *modeling framework for Internet computing*, integrating web services, messaging, ebXML, .NET and other technologies under a common technology-independent model. Please see EDOC Vision and Summary, extracted from the specification.

MDA and SOA

- **SOA per se emerged from the web services specifications of the Worldwide Web Consortium (W3C).**

- **W3C originally envisioned loosely coupled inter-enterprise B2B service components available over the Internet, exchanging information through XML documents. Such services are described in a Web Services Definition Language (WSDL) and located via a Universal Description, Discovery and Integration (UDDI) directory.**

MDA and SOA

- **SOA takes web services deeper into the enterprise, attempting to restructure a company's entire applications portfolio into a set of SOA-based service components.**

- **Components can then *hopefully* be orchestrated into new applications at will, speeding and simplifying the solutions delivery process. For the Internet-based developments, SOA *sounds like* a very intuitive extension of familiar concepts.**

MDA and SOA

- **SOA is relatively easy to describe, it is much more challenging to implement. The process of breaking down a large enterprise's business functionality and existing systems into independently deployable service components is difficult.**
- **Moreover, that process uncovers exactly the same decades-old problems that inspired the development of UML, BPMN, MOF and, ultimately, MDA.**
- **MDA, SOA *is technically just another architectural approach* to support software development.**
- **Carefully blending SOA with MDA could accelerate the successful adoption of both, and finally make clear that "real MDA" is much more than just UML-based code-generation.**

MDA Process in general– Challenges (1)

- **How to assess new requirements and change requests:**

  – how do they affect the code, the PSM, or the PIM, or CIM?

  – how to locate the model/code/transformation rule that is affected by a change request,

  - connection between CIM and PIM is tenuous at best, so the path from requirements to design is still muddy

- **In general, writing transformation rules is very difficult:**

  architects should/must understand what transformation the tool performs,

MDA Process in general– Challenges (2)

- **Transformation language not a standard yet – experiments such as project VIDE (PJWSTK a partner in an EU project) and empirical evaluations are challenging,**
- **Architects are not used to creating PIM-level design that is complete and consistent – how to proceed with those phases to get high quality outcomes? how to validate and verify design?**
- **How to introduce quality attributes into different layers and the required transformations,**
- **Reverse engineering required to make full sense from the MDA concept**

Conclusions

- **The main purpose of the MDA approach is to shorten development time and cost by easing the programming burden.**
- **In MDA, most source code is generated from the PSMs and generated by the transform tool.**
- **MDA can be an efficient development tool for cross-platform development in enterprise systems.**
- **Much more research work, experimentation and experience from practical use attempts are required to fully develop/assess the MDA idea.**

Readings

- **Alan Brown's three-part "Introduction to Model Driven Architecture" (developerWorks, February 2004 through May 2005) explains the importance of the modeling approach and introduces four key principles of MDA, as well as discussing related tools and standards.**

- **The Rational Edge has published the entire first chapter of *MDA Distilled: Principles of Model-Driven Architecture* (Addison-Wesley, 2004), an excellent resource for architects interested in learning more about MDA.**

- **In "The role of the service-oriented architect" (The Rational Edge, May 2003) Jason Bloomberg contrasts model-driven and service-oriented architectures and shows you how to apply the 4+1 view model in the design of a service-oriented architecture.**
- **Visit IBM's SOA and Web Services technical library to learn more about service-oriented architectures.**
- **The Object Management Group hosts an MDA homepage where you can read MDA success stories and learn about MDA tools and products.**